AIOLOS: mobile middleware for adaptive offloading

Tim Verbelen*, Pieter Simoens*†, Filip De Turck* and Bart Dhoedt*
* Ghent University - IBBT, Department of Information Technology
Gaston Crommenlaan 8/201, 9050 Gent, Belgium
† Ghent University College, Department INWE
Valentin Vaerwyckweg 1, 9000 Gent, Belgium
{tim.verbelen},{pieter.simoens},{filip.deturck},{bart.dhoedt}@intec.ugent.be

ABSTRACT

As the popularity of smartphones and tablets increases, the mobile platform is becoming a very important target for application developers. Despite recent advances in mobile hardware, most mobile devices still fall short to execute complex multimedia applications such as image processing. Cyber foraging is a well known computing technique to enhance the capabilities of mobile devices, where the mobile device offloads parts of the application to a nearby discovered server in the network.

In this poster, we present AIOLOS: a mobile middleware system for adaptive offloading that identifies candidate methods for offloading to the application developer. We present a model taking into account server resources and network state to decide at runtime whether or not the method execution should be offloaded. A prototype implementation of the middleware on the Android mobile platform is presented and evaluated using a real life application scenario.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Miscellaneous

General Terms

Management

Keywords

Cyber Foraging, Mobile Middleware

1. INTRODUCTION

The past decade has witnessed an increasing popularity of smartphones. According to Gartner this trend will continue, predicting an increase of 57.7 percent on smartphone sales in 2011 [2]. Advances in mobile hardware (e.g. display resolution and CPU power) and the introduction of application markets on various mobile platforms, have resulted in a myriad of mobile applications such as social and messaging clients, location-based services, games and many more.

Middleware Posters'2011, December 12th, 2011, Lisbon, Portugal.

Although the capabilities of mobile devices are increasing, they still cannot match their desktop counterparts, especially for complex multimedia applications such as image and video processing, object or face recognition and augmented reality applications. This encourages the use of cyber foraging to augment the capabilities of the mobile device by outsourcing CPU intensive parts of the application to nearby discovered resources [6], called surrogates. The biggest challenge is then to decide when and which parts of the application to outsource.

Other cyber foraging frameworks such as Scavenger [4] or MAUI [1] use a history-based profile to offload method calls. Scavenger offloads Python methods based on the size of their arguments and an estimate of the surrogate's resources, while the network is assumed as a constant. MAUI offloads methods in the .Net framework using an integer linear programming solver to take the decisions, but does not take into account the effect of input arguments on the execution time.

In this paper we present AIOLOS[1]: a mobile middleware platform for adaptive offloading built on OSGi [7], which takes both the argument size and network variability into account. For each offloadable method the middleware estimates the execution time based on the argument size, previous invocations and a network and surrogate estimation model, choosing the most appropriate execution location. Afterwards the estimation model is updated, taking into account the new profiling data. This way the system will optimize the application execution, taking into account the device capabilities and the network connectivity, without putting additional burden on the application developer. To show the effectiveness of our approach we evaluated or framework on Android using a Photo Editor application.

2. MIDDLEWARE ARCHITECTURE

The middleware presented in this paper is based on OSGi [7], a service oriented module management system in Java allowing to dynamically load and unload software modules – called bundles – at runtime. The portability of Java enables the execution of the same code on different platforms and architectures, enabling remote execution and code migration. OSGi bundles can expose a service interface by registering an implementation of this interface with the OSGi service registry, which can be queried by other bundles.

Each service interface contains a number of service methods, which will be monitored and profiled at runtime by

[1] *aiolos* is an ancient greek word meaning "quickly changing, adapting"

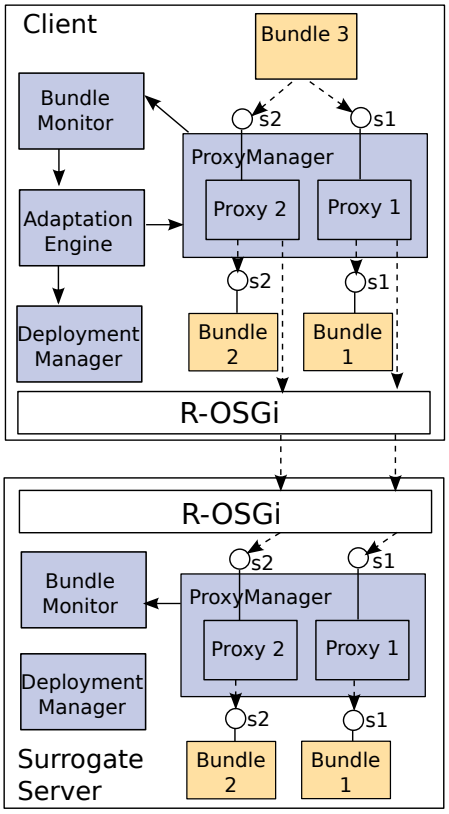

Figure 1: Architecture of the AIOLOS framework.

the middleware. Using this method profile the middleware will decide for each method call whether it will be executed locally or remotely. The architecture of the proposed middleware is shown in Figure 1. Both client and server devices run the R-OSGi [5] bundle, which enables remote method invocations across OSGi platforms. The remote server can be discovered using a service discovery protocol such as SLP [3] or by direct connection to a known address.

The Proxy Manager is responsible for creating and managing proxies for each application bundle's services. By proxying each of those registered services, the middleware can on the one hand decide to forward a service call to a remote service implementation rather than to the local one, and on the other hand gather monitoring information of all service calls.

The Bundle Monitor gathers monitor information of all application bundles. It gets notified by the proxies when a service method is called, and captures the size of the input arguments, the size of the return value and the execution time of the method call. Using this information, an execution profile for each method can be constructed, which is used to identify methods for remote execution by the Offloading Logic bundle.

Offloading Logic provides the algorithm to decide whether to offload a method or not. Using the monitor information of the Bundle Monitor it will estimate the parameters of the network and the capabilities of the server. A method call is offloaded if its estimated remote execution time T_{remote} is smaller than its estimated local execution time T_{local}, using the following formulas.

$$\hat{T}_{local} = f(I) \quad (1)$$

$$\hat{T}_{remote} = \alpha \hat{T}_{local} + \beta(I + g(I)) + \gamma \quad (2)$$

T_{local} is estimated in function of the input parameters I using history. T_{remote} is estimated using a speedup factor α of the server, and network parameters β and γ representing the network round trip time, which increases in function of the bytes sent, i.e. the input parameters I and estimated return value $g(I)$ which size is also estimated from history.

The Deployment Manager copies bundles that would benefit from remote execution by sending the bytecodes to the server side and installing the bundle there on the OSGi runtime.

3. PERFORMANCE

For evaluation we have integrated the AIOLOS framework in the Android OS and performed experiments with a Photo Editor application. Using the Photo Editor application the user can perform image filters such as a blur or sharpen filter on the pictures on his device. Experiments on a Motorola Droid device show that the AIOLOS framework is able to correctly estimate the local and remote execution time. Depending on the network latency, the AIOLOS framework will only offload method calls on images large enough to benefit from remote execution.

4. ACKNOWLEDGMENTS

Tim Verbelen is funded by Ph.D grant of the Fund for Scientific Research, Flanders (FWO-V).

5. REFERENCES

[1] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62, New York, NY, USA, 2010. ACM.

[2] Gartner Group. 2011 press releases. http://www.gartner.com/it/page.jsp?id=1622614.

[3] E. Guttman. Service location protocol: Automatic discovery of ip network services. *IEEE Internet Computing*, 3:71–80, July 1999.

[4] M. D. Kristensen and N. O. Bouvin. Scheduling and development support in the scavenger cyber foraging system. *Pervasive and Mobile Computing*, 6(6):677–692, 2010. Special Issue PerCom 2010.

[5] J. S. Rellermeyer, G. Alonso, and T. Roscoe. R-osgi: distributed applications through software modularization. In *Middleware '07: Proceedings of the International Conference on Middleware*, pages 1–20, New York, NY, USA, 2007. Springer-Verlag New York, Inc.

[6] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8:10–17, 2001.

[7] The OSGi Alliance. *OSGi Service Platform, Core Specification, Release 4, Version 4.2*. aQute, September 2009.